600.363/463 Algorithms - Fall 2013
Solution to Assignment 9

(50 points)

34.2-1 Consider the language GRAPH-ISOMORPHISM= $\{\langle G_1, G_2\rangle : G_1$ and G_2 are isomorphic graphs$\}$. Prove that GRAPH-ISOMORPHISM $\in$ NP by describing a polynomial-time algorithm to verify the language.

Proof. We aim to show that the language GRAPH-ISOMORPHISM can be verified in polynomial time. Let the input x be two graphs G_1 and G_2 and let the certificate y be the indices $\{i_1, i_2, \cdots, i_n\}$. An algorithm $A(x, y)$ verifies GRAPH-ISOMORPHISM by executing the following steps:

(a) Check if the certificate y is a permutation of $\{1, 2, \cdots, n\}$. If no, return false; else continue.

(b) Permute the vertices of G_1 as given by the given permutation. Verify that the permuted G_1 is identical to G_2.

Step (a) takes at most $O(V^2)$ time and step (b) runs in $O(V + E)$ time, therefore the verification algorithm A runs in $O(V^2)$ time. GRAPH-ISOMORPHISM $\in$ NP. □

34.2-6 A ***Hamiltonian path*** in a graph is a simple path that visits every vertex exactly once. Show that the language HAM-PATH $= \{\langle G, u, v\rangle :$ there is a hamiltonian path from u to v in graph $G\}$ belongs to NP.

Proof. We aim to show that the language HAM-PATH can be verified in polynomial time. Let the input x be $\langle G, u, v\rangle$ and let the certificate y be a sequence of vertices $\{v_1, v_2, \cdots, v_n\}$. An algorithm $A(x, y)$ verifies HAM-PATH by executing the following steps:

(a) Check if $|G.V| = n$;

(b) Check if $v_1 = u$ and $v_n = v$;

(c) Check if $\forall i \in \{1, 2, \cdots, n\}$, $v_i \in G.V$;

(d) Check if $\forall i, j \in \{1, 2, \cdots, n\}$, $v_i \neq v_j$;

(e) Check if $\forall i \in \{1, 2, \cdots, n-1\}$, $(v_i, v_{i+1}) \in G.E$;

If any of the above step fails, return false. Else return True;
Steps (a) and (b) takes $O(1)$ time; step (c) takes $O(V)$ time; step (d) runs in $O(V^2)$ time and step (e) runs in $O(E)$ time. Therefore the verification algorithm runs in $O(V^2)$ time. Hence HAM-PATH $\in$ NP. □

34.3-2 Show that the $\leq_P$ relation is a transitive relation on languages. That is, show that if $L_1 \leq_P L_2$ and $L_2 \leq_P L_3$, then $L_1 \leq_P L_3$.

Proof. Let $L_1 \leq_P L_2$ and $L_2 \leq_P L_3$, i.e. there exist polynomial-time computable reduction functions $f_1 : \{0,1\}^* \to \{0,1\}^*$ and $f_2 : \{0,1\}^* \to \{0,1\}^*$ such that

$$x \in L_1 \Leftrightarrow f_1(x) \in L_2$$
$$x \in L_2 \Leftrightarrow f_2(x) \in L_3$$

Define $f_3 = f_1 \circ f_2$, then L_3 is a polynomial-time computable function : $\{0,1\}^* \to \{0,1\}^*$, and

$$x \in L_1 \Leftrightarrow f_3(x) \in L_3$$

holds. Hence $L_1 \leq_P L_3$. □

34.4-5 Show that the problem of determining the satisfiability of boolean formulas in disjunctive normal form is polynomial-time solvable.

Proof. A boolean formula in disjunctive normal form(DNF) is composed of OR of clauses of ANDs. It is satisfiable if any of its clause can be evaluated 1. If there is some variable x such that one clause contains $x \wedge \neg x$, the clause will evaluate 0 whatever boolean value x is assigned; else there is some assignment of values 0 and 1 to its variable that causes it to evaluate 1. Therefore the determining algorithm simply checks every clause and will return false there is some variable x such that x and $\neg x$ exist in the same clause. Else the algorithm will return true.
Assume that there are m clauses and each clause has at most k literals. Therefore each clause can be evaluated in $O(k^2)$ time and the whole algorithm runs in $O(mk^2)$. Hence the problem of determining the satisfiability of boolean formulas in disjunctive normal form is polynomial-time solvable. □

34.4-6 Suppose that someone gives you a polynomial-time algorithm to decide formula satisfiability. Describe how to use this algorithm to find satisfying assignments in polynomial time.

Solution. The language for formula satisfiability problem is SAT= $\{\langle\phi\rangle : \phi$ is a satisfiable boolean formula$\}$. Let ϕ be a boolean formula and A be the polynomial-time algorithm to decide the SAT. In addition assume ϕ has at most n variables, denoted as $\{x_1, x_2, \cdots, x_n\}$. Denote $\phi(x_i = 0)$ and $\phi(x_i = 1)$ as the formulas when a variable $x_i, i \in \{1, 2, \cdots, n\}$ is assigned 0 or 1, respectively. The following procedure can find a satisfying assignment:

(a) Apply A on ϕ to decide whether ϕ is satisfiable. If NO, return "no satisfying assignment can be found"; else continue.

(b) Initialize $i = 1$.

(c) Let $x_i = 0$, then $\phi(x_i = 0)$ is a boolean formula with $n - i$ variables. Apply A to check the satisfiability of $\phi(x_i = 0)$. If it returns YES, record $x_i = 0$. Else record $x_i = 1$. Since ϕ is satisfiable, there must be an assignment of x_i satisfies ϕ;

(d) Replace x_i in ϕ by the recorded value from step (c), then increase i by 1, repeat step (c) until $i = n$;

(e) return the recorded assignments for all variables $\{x_1, x_2, \cdots, x_n\}$.

Since A runs in polynomial time, and the above procedure calls A n times, hence it runs in polynomial time. □